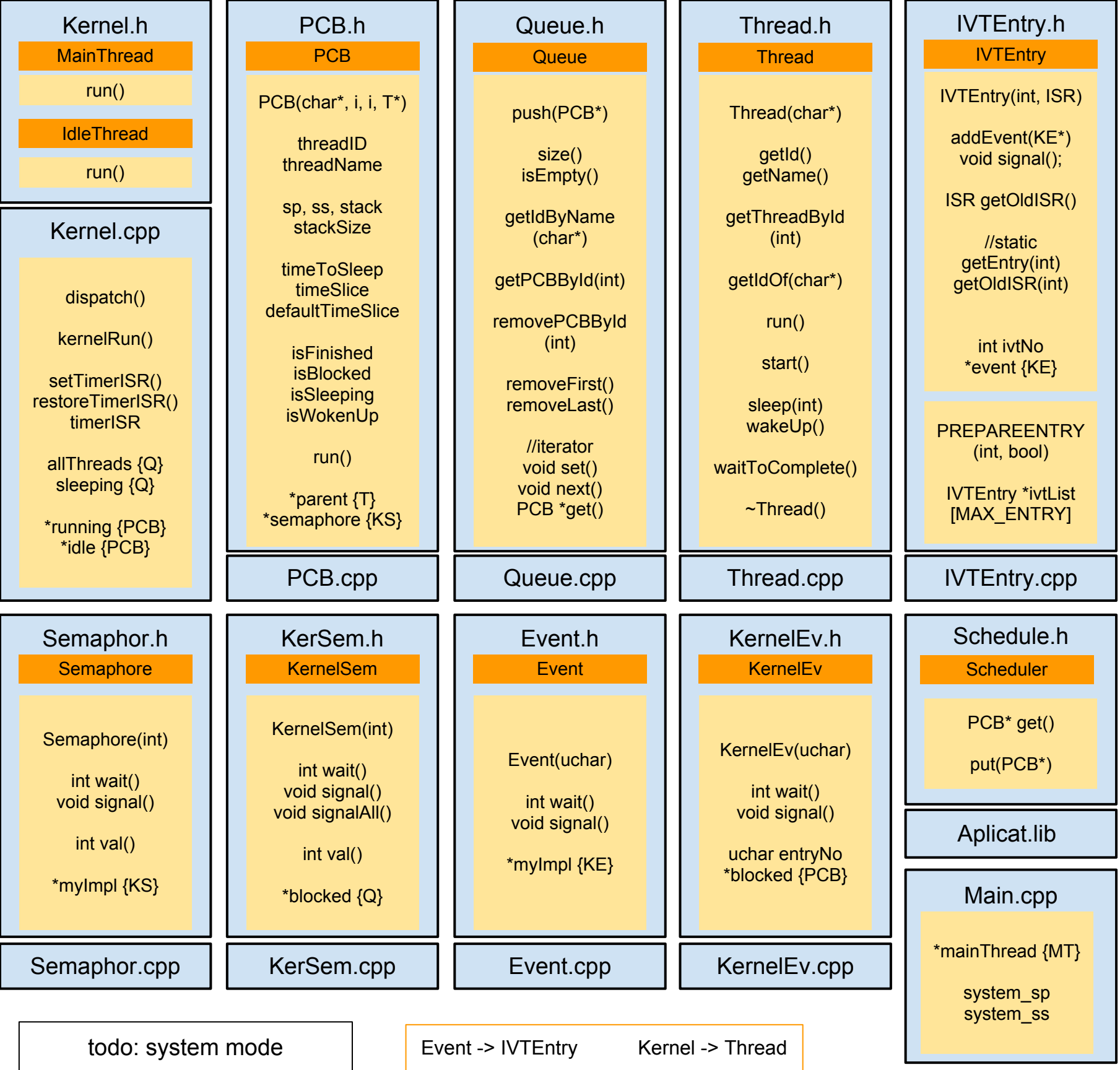
Kernel.h
MainThread
run()
IdleThread
run()
Kernel.cpp
dispatch()
kernelRun()
setTimerISR()
restoreTimerISR()
timerISR
allThreads {Q}
sleeping {Q}
*running {PCB}
*idle {PCB}
PCB.h
PCB
PCB(char*, i, i, T*)
threadID
threadName
sp, ss, stack
stackSize
timeToSleep
timeSlice
defaultTimeSlice
isFinished
isBlocked
isSleeping
isWokenUp
run()
*parent {T}
*semaphore {KS}
PCB.cpp
Queue.h
Queue
push(PCB*)
size()
isEmpty()
getIdByName
(char*)
getPCBById(int)
removePCBById
(int)
removeFirst()
removeLast()
//iterator
void set()
void next()
PCB *get()
Queue.cpp
Thread.h
Thread
Thread(char*)
getId()
getName()
getThreadById
(int)
getIdOf(char*)
run()
start()
sleep(int)
wakeUp()
waitToComplete()
~Thread()
Thread.cpp
IVTEntry.h
IVTEntry
IVTEntry(int, ISR)
addEvent(KE*)
void signal();
ISR getOldISR()
//static
getEntry(int)
getOldISR(int)
int ivtNo
*event {KE}
PREPAREENTRY
(int, bool)
IVTEntry *ivtList
[MAX_ENTRY]
IVTEntry.cpp
Semaphor.h
Semaphore
Semaphore(int)
int wait()
void signal()
int val()
*myImpl {KS}
Semaphor.cpp
KerSem.h
KernelSem
KernelSem(int)
int wait()
void signal()
void signalAll()
int val()
*blocked {Q}
KerSem.cpp
Event.h
Event
Event(uchar)
int wait()
void signal()
*myImpl {KE}
Event.cpp
KernelEv.h
KernelEv
KernelEv(uchar)
int wait()
void signal()
uchar entryNo
*blocked {PCB}
KernelEv.cpp
Schedule.h
Scheduler
PCB* get()
put(PCB*)
Aplicat.lib
Main.cpp
*mainThread {MT}
system_sp
system_ss
todo: system mode
Event -> IVTEntry
Kernel -> Thread